Compact Ancestry Labeling Schemes for Trees of Small Depth*

Pierre Fraigniaud
CNRS and Univ. Paris Diderot
pierre.fraigniaud@liafa.jussieu.fr

Amos Korman
CNRS and Univ. Paris Diderot
amos.korman@liafa.jussieu.fr

arXiv:0902.3081v1 [cs.DS] 18 Feb 2009

Abstract

An *ancestry labeling scheme* labels the nodes of any tree in such a way that ancestry queries between any two nodes in a tree can be answered just by looking at their corresponding labels. The common measure to evaluate the quality of an ancestry labeling scheme is by its *label size*, that is the maximal number of bits stored in a label, taken over all n-node trees. The design of ancestry labeling schemes finds applications in XML search engines. In the context of these applications, even small improvements in the label size are important. In fact, the literature about this topic is interested in the exact label size rather than just its order of magnitude. As a result, following the proposal of an original scheme of size $2\log n$ bits, a considerable amount of work was devoted to improve the bound on the label size. The current state of the art upper bound is $\log n + O(\sqrt{\log n})$ bits which is still far from the known $\log n + \Omega(\log\log n)$ lower bound. Moreover, the hidden constant factor in the additive $O(\sqrt{\log n})$ term is large, which makes this term dominate the label size for typical current XML trees.

In attempt to provide good performances for real XML data, we rely on the observation that the depth of a typical XML tree is bounded from above by a small constant. Having this in mind, we present an ancestry labeling scheme of size $\log n + 2\log d + O(1)$, for the family of trees with at most n nodes and depth at most d. In addition to our main result, we prove a result that may be of independent interest concerning the existence of a linear *universal graph* for the family of forests with trees of bounded depth.

*This research is supported in part by the ANR project ALADDIN, by the INRIA project GANG, and by COST Action 295 DYNAMO.

1 Introduction

1.1 Background

It is often the case that when people wish to retrieve data from the Internet, they use search engines like Yahoo or Google which provide full-text indexing services (the user gives some keywords and the engine returns documents containing these keywords). In contrast to such search engines, the evolving XML Web-standard [2, 33] aims for allowing more sophisticated queries of documents. By describing the semantic structure of the document components, it allows users to not only ask full-text queries (find documents containing the phrase "computer science researches") but also ask for more sophisticated data (find all items about computer science researches that did their Phd at ETH Zürich and whose age is below 35).

To implement such sophisticated queries, Web documents obeying the XML standard are viewed as labeled trees, and typical queries over the documents amount to testing relationships between document items, which correspond to ancestry queries among the corresponding tree nodes [2, 8, 34, 35]. To process such queries, XML query engines often use an index structure, typically a big hash table, whose entries are the tag names in the indexed documents. Due to the enormous size of the Web data and to its distributed nature, it is essential to answer queries using the index labels only, without accessing the actual documents. To allow good performances, it is essential that a large portion of the index structure resides in the main memory. Since we are dealing here with a huge number of index labels, reducing the length of the label size, even by a constant factor, is critical for the reduction of memory cost and for performance improvement. For more details regarding XML search engines see, e.g., [1, 3, 7].

Labeling schemes which are currently being used by actual systems are variants of the following interval based ancestry labeling scheme [16, 30]. Enumerate the leaves from left to right and label each node u by the interval $[\ell_s, \ell_l]$, where ℓ_s (respectively, ℓ_l) is the smallest (resp., largest) leaf descendant of u. An ancestry query then amounts to an interval containment query between the corresponding interval labels. It is easy to see that the size of the labels produced by this simple scheme is bounded by $2\log n$ bits, where n is the size of the tree.

A considerable amount of research was devoted to improve the upper bound on the label size as much as possible [1, 3, 32]. The current state of the art upper bound [1] is $\log n + O(\sqrt{\log n})$ which is still far from the known $\log n + \Omega(\log\log n)$ lower bound [4]. Moreover, the hidden constant factor in the additive $O(\sqrt{\log n})$ term is large, which makes this term dominate the label size in the average size of current applications. Following that work, [15] suggested other ancestry labeling schemes whose worst case bound is $1.5\log n + O(1)$ but perform better than the scheme of [1] for typical XML data.

In attempt to provide good performances for real XML instances, we rely on the observation that a typical XML tree has extremely small depth (cf. [7, 26, 25]). For example, by examining about 200,000 XML documents on the Web, Mignet et al. [25] found that the average depth of an XML tree is 4, and that 99% of the trees have depth at most 8. Motivated by this observation, we concentrate on bounded depth trees, and prove an upper bound of $\log n + 2\log d + O(1)$ for the size of an ancestry labeling scheme for the family of n-node trees whose depth is bounded by d. (In fact, our bound holds even for forests rather than just for trees.)

It is not clear whether one can adapt the techniques from previous schemes to perform better on trees of small depth. For example, the simple interval scheme has label size $2\log n$ also for trees with constant depth. As another example, before starting the actual labeling process, the ancestry scheme in [1] first transforms the given tree to a binary tree. This transformation

already results with a tree of depth $\Omega(\log n)$, even if the given tree has constant depth. Moreover, previous relevant schemes extensively use and rely on a specific technique, for using *alphabetic codes* on different subpaths. This technique, at least on its surface, does not seem to be more effective on short subpaths, than on long ones.

In contrast, this paper uses a different technique that does not rely on alphabetic codes. Informally, the idea behind our scheme is the following. The labels of the nodes are taken from a small set of integers U, thus ensuring short labels. Each integer in U is associated with some interval taken from some limited range. The fundamental rule of our labeling scheme is that a node u is an ancestor of v if and only if the interval associated with the label of u (i.e., the corresponding integer in U) contains the interval associated with the label of v. That way, the ancestry query can be answered very easily, simply by comparing the corresponding intervals. The main technical challenge is to find a way to define and nest these intervals between themselves to be able to appropriately map the nodes of any n-node forest of bounded depth into U, while keeping U small.

1.2 Other related work

Implicit labeling schemes were first introduced in [16], where an elegant adjacency labeling schemes of size $2\log n$ is established on n-node trees. That paper also notices a relation between adjacency labeling schemes and *universal graphs* (see also [6, 10, 24]). Precisely, it is shown that there exists an adjacency labeling scheme with label size k for a graph family $\mathcal{G}$ if and only if there exists a universal graph for $\mathcal{G}$ with 2^k nodes.

Adjacency labeling schemes on trees were further investigated in an attempt to reduce the constant factor in the label size. In [14] an adjacency labeling scheme using label size of $\log n + O(\sqrt{\log n})$ is presented; and in [6] the label size was further reduced to $\log n + O(\log^* n)$. This current state of the art bound implies the existence of a universal graph for the family of n-node trees with $2^{O(\log^*(n))}n$ nodes.

Labeling schemes were also proposed for other decision problems on graphs, including distance [4, 10, 12, 13, 14, 24, 27, 31], routing [9, 32], flow [21, 17], vertex connectivity [19, 17], nearest common ancestor [5, 28], and various other tree functions, such as center, separation level, and Steiner weight of a given subset of vertices [28]. See [11] for a survey on static labeling schemes. Dynamic labeling schemes were investigated in a number of papers, e.g., [18, 20, 23, 22].

1.3 Our contributions

We present an ancestry labeling scheme of size $\log n + 2\log d + O(1)$ for the family of rooted forests with at most n nodes and depth at most d. Our result is essentially optimal for rooted trees with constant depth, and thus for the typical XML trees.

As a corollary of our main theorem, we get an adjacency scheme of size $\log n + 3\log d + O(1)$ for the family of forests with at most n nodes and depth bounded by d. This, in particular, implies the existence of a linear universal graph for the family of forests with constant depth. Namely, we show the existence of a graph of size $O(n)$ that contains all n-node forests of constant depth as vertex induced subgraphs.

2 Preliminaries

Let T be a tree rooted at some node r referred as the *root* of T. The *depth* of a node $u \in V(T)$ is defined as 1 plus the hop distance from u to the root of T. In particular, the depth of the root is 1. The depth of T is the maximum depth of a node in T. Let u and v be two nodes in T. We say that u is an *ancestor* of v if $u \neq v$ and u is one of the nodes on the shortest path connecting v and the root of T.

A *rooted forest* F is a collection of rooted trees. The depth of F is the maximum depth of a tree in F. For two nodes u and v in F, we say that u is an ancestor of v if and only if u is an ancestor of v in one of the trees in F. For integers n and d, let $\mathcal{F}(n, d)$ denote the family of all rooted forests with at most n nodes and depth bounded from above by d.

An *ancestry labeling scheme* $(\mathcal{M}, \mathcal{D})$ for a family of rooted forests $\mathcal{F}$ is composed of the following components:

1. A *marker* algorithm $\mathcal{M}$ that, given a forest F in $\mathcal{F}$, assigns labels to its nodes.
2. A polynomial time *decoder* algorithm $\mathcal{D}$ that given two labels ℓ_1 and ℓ_2 in the output domain of $\mathcal{M}$, returns a boolean in $\{0, 1\}$.

These components must satisfy that if $L(u)$ and $L(v)$ denote the labels assigned by the marker to two nodes u and v in some rooted forest $F \in \mathcal{F}$, then

$$\mathcal{D}(L(u), L(v)) = 1 \iff u \text{ is an ancestor of } v \text{ in } F.$$

It is important to note that the decoder $\mathcal{D}$ is independent of the forest F. Thus $\mathcal{D}$ can be viewed as a method for computing ancestry values in a "distributed" fashion, given any pair of labels and knowing that the forest belongs to some specific family $\mathcal{F}$.

The common complexity measure used to evaluate a labeling scheme $(\mathcal{M}, \mathcal{D})$ is the *label size*, that is the maximum number of bits in a label assigned by the marker algorithm $\mathcal{M}$ to any node in any forest in $\mathcal{F}$.

Given two integers a and b, where $a < b$, let $[a, b]$ (respectively, $[a, b)$) denote the interval containing the integers i such that $a \leq i \leq b$ (resp., $a \leq i < b$). Given a graph G, let $|G|$ denote the number of nodes in G.

3 A compact ancestry labeling scheme for $\mathcal{F}(n, d)$

This section is devoted to proving the existence of an ancestry labeling scheme of size $\log n + 2 \log d + O(1)$ for the family of rooted forests in $\mathcal{F}(n, d)$. Informally, the scheme performs as follows. We construct a set of intervals U such that the nodes of any forest in $\mathcal{F}(n, d)$ can be mapped to U, in a way that ancestry relation can be answered using a simple interval containment test. I.e., we make sure that u is an ancestor of v in some forest F if and only if the interval associated with u contains the interval associated with v. We call such a mapping an *ancestry mapping*. A label of a node in F is simply a pointer to an element in U, and thus can be encoded using $\log |U|$ bits. Therefore, to get short labels we need U to be small.

The construction of U is done by induction on the number of nodes in the forest. Assume that there exists some set of intervals U_k, such that for any forest of size at most 2^k, there exists an ancestry mapping from F to U_k, and consider now the set of forests $\mathcal{F}_{k+1}$ with at most 2^{k+1}

nodes. Of course, if every $F \in \mathcal{F}_{k+1}$ would break nicely into two forests with at most 2^k nodes each, then one could embed the two parts separately on two interval sets U' and U'' of the same size as U_k. If that was always the case, we would ultimately get an interval set $U = U_{\log n}$ of linear size for which any forest of size at most n could be embedded to U via an ancestry mapping, and that would yield an ancestry labeling scheme with label size $\log n$.

Fortunately, life is not so simple, and a forest $F \in \mathcal{F}_{k+1}$ doesn't always break nicely to two equal size sub-forests. Specifically, problems occur whenever one must break a tree T of F into two parts and embed one part in U' and the other part in U''. Ideally, if F is broken into $F' \cup F'' \cup T$ where F' is embedded in $I' \subseteq U'$, and F'' is embedded in $I'' \subseteq U''$, then one wants to embed T by borrowing what remains free in $U' \setminus I'$ and $U'' \setminus I''$. This can be achieved by using various scales of sub-interval sizes, so that to embed T in $J = J' \cup J''$ with $J' \subseteq U' \setminus I'$ and $J'' \subseteq U'' \setminus I''$.

Two difficulties arise in this recursive approach. The first one is related to the scale of the sub-intervals in which one picks J' and J''. Indeed, too many sub-intervals yields too many intervals in U_{k+1}. On the other hand, too few sub-intervals yields too large gaps between I' and J' in U'. This prevents the intervals in U_{k+1} from being sufficiently compressed, and thus also ultimately results with too many intervals in U_{k+1}. Determining a good tradeoff between the amount of scaling in the sub-intervals, and the gaps between intervals, in thus one major issue.

The second difficulty that is faced by the recursive approach is that splitting a tree into subtrees of sizes at most half is performed by removing the separator of the tree. However, one can see that whenever a tree T of 2^{k+1} nodes is split into a collection $T_1, \ldots, T_\ell$ of subtrees by removing the separator of T, the subtree containing the root of T plays a special role in the setting of the ancestry scheme. Dealing with this special subtree is a second important issue, for which the assumption on the depth of the forests will play a major role. The proof of the theorem below shows how to overcome these two issues.

Theorem 3.1 *There exists an ancestry labeling scheme for the family of rooted forests in $\mathcal{F}(n,d)$ whose label size is $\log n + 2\log d + O(1)$.*

Proof. For simplicity, we assume n is a power of 2. (If n is not a power of 2, we just round it to the next power of 2, say N, and we add $N - n$ independent nodes to the forest). We begin by defining a set $U = U(n,d)$ of integers, which we use later to label all forests in $\mathcal{F}(n,d)$.

Let $c_0 = 1$, and, for any i, $1 \leq i \leq \log n$, let

$$c_i = c_{i-1} + 1/i^2 = 1 + \sum_{j=1}^{i} 1/j^2 \ .$$

We have $1 + \sum_{j \geq 1} 1/j^2 \leq 3$, and hence all the c_i's are bounded from above by 3. For any i, $1 \leq i \leq \log n$, let us define the following values, that will be used to decompose integers:

$$\begin{aligned} H_i &= 1 + 3 \cdot n \cdot d \cdot i^2/2^{i-1} \\ J_i &= 2 \cdot d \cdot c_i \cdot i^2 \end{aligned}$$

Then we define $\Gamma_0 = 3n$, and

$$\Gamma_i = \Gamma_0 + \sum_{j=1}^{i} H_j \cdot J_j.$$

The set of integers U is defined as the interval

$$U = [1, \Gamma_{\log n}).$$

Note that since $\Gamma_{\log n} = O(nd^2)$, we have $|U| = O(nd^2)$. The marker algorithm maps the nodes of any forest $F \in \mathcal{F}(n,d)$ into the integer set U. To perform, the decoder algorithm represents each integer in U as a unique triplet (i,h,j), as follows.

- An integer $\nu \in [1, \Gamma_0)$ is simply represented by $(0, \nu, 0)$;
- An integer ν that satisfies $\Gamma_{i-1} \leq \nu < \Gamma_i$ for some $1 \leq i \leq \log n$ can be described as

$$\nu = \Gamma_{i-1} + hJ_i + j$$

  for unique h and j such that $h \in [0, H_i)$ and $j \in [0, J_i)$; Hence we represent such ν by the triplet (i,h,j).

For simplicity of presentation, in the following, we will not distinguish between an integer in U and its triplet representation, unless it may cause a confusion. Every integer in U is associated with an interval as follows. Let $x_0 = 1$, and for any i, $1 \leq i \leq \log n$, let

$$x_i = \left\lceil \frac{2^{i-1}}{di^2} \right\rceil .$$

For $h \in [0, \Gamma_0)$, we associate the triplet $(0,h,0) \in U$ with the interval $I_{0,h,0} = [h]$. For any i, $1 \leq i \leq \log n$, any $h \in [0, H_i)$, and any $j \in [0, J_i)$, we associate the triplet $(i,h,j) \in U$ with the interval

$$I_{i,h,j} = [x_i h,\ x_i(h+j)).$$

We now define a concept of specific interest for the purpose of our proof:

Definition 1 *Let $F \in \mathcal{F}(n,d)$. We say that a mapping $L : F \to U$ is an* ancestry mapping *if, for every two nodes $u, v \in F$ with $L(u) = (i,h,j)$ and $L(v) = (i',h',j')$, we have*

$$u \text{ is an ancestor of } v \text{ in } F \iff I_{i',h',j'} \subseteq I_{i,h,j}.$$

In order to show that there exists an ancestry mapping from every forest in $\mathcal{F}(n,d)$ into U, we shall make use of the following definitions. For any interval $I \subseteq [1, \Gamma_0)$, let

$$U_0(I) = \{(0, \nu, 0) \mid \nu \in I\}$$

and, for any k, $1 \leq k \leq \log n$, let

$$U_k(I) = U_0(I) \cup \{(i,h,j) \mid 1 \leq i \leq k,\ h \in [0, H_i),\ j \in [0, J_i) \text{ and } I_{i,h,j} \subseteq I\}.$$

The following observations are immediate by the definition of the sets $U_k(I)$. Let I and J be two intervals in $[1, \Gamma_0)$. For any k, $1 \leq k \leq \log n$, we have:

- $I \cap J = \emptyset \ \Rightarrow\ U_k(I) \cap U_k(J) = \emptyset$,
- $U_k(I) \cup U_k(J) \subseteq U_k(I \cup J)$,
- $I \subset J \ \Rightarrow\ U_k(I) \subset U_k(J)$,
- $U_{k-1}(I) \subset U_k(I)$.

Fix k such that $0 \leq k \leq \log n$. We now give a sufficient condition for the existence of an ancestry labeling scheme using labels in $U_k(I)$. Let I be an interval in $[1, \Gamma_0)$ and let $I_1, I_2, \cdots, I_t$ be a partition of I into t disjoint intervals, i.e., $I = \cup_{i=1}^{t} I_i$ with $I_i \cap I_j = \emptyset$ for any $1 \leq i < j \leq t$. Let F be a forest, and let $F_1, F_2, \cdots, F_t$ be t pairwise disjoint forests such that $\cup_{i=1}^{t} F_i = F$. Using the four properties listed above, one can easily prove the following.

Claim 3.2 *If there exists an ancestry mapping from F_i to $U_k(I_i)$ for every i, $1 \leq i \leq t$, then there exists an ancestry mapping from F to $U_k(I)$.*

The following is the main technical ingredient for proving the theorem.

Claim 3.3 *For every k, $0 \leq k \leq \log n$, every forest F of size $|F| \leq 2^k$ with depth bounded by d, and every interval $I \subseteq [1, \Gamma_0)$, such that $|I| = \lfloor c_k |F| \rfloor$, there exists an ancestry mapping of F into $U_k(I)$.*

We prove this claim by induction on k. The claim for $k = 0$ holds trivially. Assume now that the claim holds for k with $0 \leq k < \log n$, and let us show that it also holds for $k+1$.

Let F be a forest of size $|F| \leq 2^{k+1}$, and let $I \subseteq [1, \Gamma_0)$ be an interval, such that $|I| = \lfloor c_{k+1} |F| \rfloor$. Our goal is to show that there exists an ancestry mapping of F into $U_{k+1}(I)$. We consider two cases.

The simpler case is when all the trees in F are of size at most 2^k. For this case, we show a claim stronger than what is stated in Claim 3.3. Specifically, we show that there exists an ancestry mapping of F into $U_k(I)$ for every interval $I \subseteq [1, \Gamma_0)$ such that $|I| = \lfloor c_k |F| \rfloor$ (i.e., a fraction $1/(k+1)^2$ of $|F|$ smaller than what is required to prove the claim). Let $T_1, T_2, \cdots T_\ell$ be the trees in F. We divide the given interval I of size $\lfloor c_k |F| \rfloor$ into $\ell + 1$ disjoint subintervals $I = I_1 \cup I_2 \cdots \cup I_\ell \cup I'$, where $|I_i| = \lfloor c_k |T_i| \rfloor$ for every i, $1 \leq i \leq \ell$. This can be done because $\sum_{i=1}^{\ell} \lfloor c_k |T_i| \rfloor \leq \lfloor c_k |F| \rfloor = |I|$. By the induction hypothesis, we have an ancestry mapping of T_i into $U_k(I_i)$ for every i, $1 \leq i \leq \ell$. The stronger claim thus follows in this case by Claim 3.2.

Now consider the more involved case in which one of the subtrees in F, denoted by T^*, contains more then 2^k nodes. Our goal now is to show that for every interval $I^* \subseteq [1, \Gamma_0)$, where $|I^*| = \lfloor c_{k+1} |T^*| \rfloor$, there exists an ancestry mapping of T^* into $U_{k+1}(I^*)$. Once we show this, we can, similarly to the first case, divide the interval I into 3 disjoint subintervals

$$I = I^* \cup I' \cup I'',$$

where

$$|I^*| = \lfloor c_{k+1} |T^*| \rfloor \text{ and } |I'| = \lfloor c_k |F'| \rfloor$$

with $F' = F \setminus T^*$. Since we have an ancestry mapping that maps T^* into $U_{k+1}(I^*)$, and one that maps F' into $U_k(I')$, we get the desired ancestry mapping of F into $U_{k+1}(I)$ by Claim 3.2. (The ancestry mapping of F' into $U_k(I')$ can be done by the induction hypothesis, because $|F'| \leq 2^k$).

For the rest of the proof, our goal is thus to prove the following claim: for every tree T of size $|T|$ with $2^k < |T| \leq 2^{k+1}$, and every interval $I \subseteq [1, \Gamma_0)$, where $|I| = \lfloor c_{k+1} |T| \rfloor$, there exists an ancestry mapping of T into $U_{k+1}(I)$.

Recall that a *separator* of a tree T is a node v whose removal from T (together with all its incident edges) brakes T into subtrees, each of size at most $|T|/2$. It is a well known fact that every tree has a separator. Note however, that there can be more than one separator to a tree.

Nevertheless, if this is the case then there are in fact two separators, and one is the parent of the other. In the following, whenever we consider a separator of a rooted tree T, we refer only to the separator of T which is closer to the root.

We make use of the following decomposition of T. We refer to the path S from the separator of T to the root of T as the *spine* of T. This spine may consist of only one node, namely, the root of T. Let $v_1, v_2, \cdots, v_{d'}$ be the nodes of the spine S, ordered bottom-up, i.e., v_1 is the separator of T and $v_{d'}$ is the root of T. By this definition, we have that if $1 \leq i < j \leq d'$ then v_j is an ancestor of v_i. A separator is not a leaf if $|T| > 1$, and therefore $1 \leq d' < d$. (Recall that the depth is 1 plus the distance to the root). By removing the nodes in the spine (and the edges connected to them), the tree T brakes into d' forests $F_1, F_2, \cdots, F_{d'}$, such that the following holds for each $1 \leq i \leq d'$:

- in T, the roots of the trees in F_i are connected to v_i;
- each tree in F_i contains at most 2^k nodes.

The given interval I for which we want to embed T into $U_{k+1}(I)$ can be expressed as $I = [a, b)$ for some integers a and b, and we have

$$b - a = |I| = \lfloor c_{k+1}|F| \rfloor .$$

For every $i = 1, \ldots, d'$, we now define an interval I_i (later, we will map F_i into $U_k(I_i)$). Let us first define I_1. Let h_1 be the smallest integer such that $a \leq h_1 x_{k+1}$, and let $\bar{h}_1$ be the smallest integer such that $\lfloor c_k|F_1| \rfloor \leq \bar{h}_1 x_{k+1}$. Note that $\bar{h}_1 \geq 1$. We let

$$I_1 = [h_1 x_{k+1}, (h_1 + \bar{h}_1) x_{k+1}).$$

Assume now that we have defined the interval

$$I_i = [h_i x_{k+1}, (h_i + \bar{h}_i) x_{k+1})$$

for $1 \leq i < d'$. We define the interval I_{i+1} as follows. Let $h_{i+1} = h_i + \bar{h}_i$ and let $\bar{h}_{i+1}$ be the smallest integers such that $\lfloor c_k|F_{i+1}| \rfloor \leq \bar{h}_{i+1} x_{k+1}$. We let

$$I_{i+1} = [h_{i+1} x_{k+1}, (h_{i+1} + \bar{h}_{i+1}) x_{k+1}).$$

Observe that for $1 \leq i \leq d'$, the interval I_i is simply $I_{k+1,h_i,\bar{h}_i}$. Note also that for every $i = 1, \ldots, d'$, we have

$$\bar{h}_i x_{k+1} < \lfloor c_k|F_i| \rfloor + x_{k+1}.$$

It follows that the size of I_i at most $\lfloor c_k|F_i| \rfloor + x_{k+1} - 1$. Thus, since $h_1 x_{k+1} < a + x_{k+1}$, we get that

$$\begin{aligned} \bigcup_{i=1}^{d'} I_i &\subseteq \Big[a,\ a + (d'+1)(x_{k+1} - 1) + \lfloor c_k|T| \rfloor \Big) \\ &\subseteq \Big[a,\ a + d \cdot (x_{k+1} - 1) + \lfloor c_k|T| \rfloor \Big). \end{aligned}$$

Now, since $d \cdot (x_{k+1} - 1) \leq \left\lfloor \frac{2^k}{(k+1)^2} \right\rfloor$, and $2^k < |T|$, it follows that,

$$\begin{aligned} \bigcup_{i=1}^{d'} I_i &\subseteq \left[a, a + \left\lfloor \frac{|T|}{(k+1)^2} + c_k|T| \right\rfloor \right) \\ &= [a, a + \lfloor c_{k+1}|T| \rfloor) \\ &= I. \end{aligned}$$

On the other hand, note that for $1 \leq i \leq d'$, I_i contains at least $\lfloor c_k|F_i| \rfloor$ nodes. Therefore, by the fact that, for any i, $1 \leq i \leq d'$, each tree in F_i contains at most 2^k nodes, we get that there exists an ancestry mapping of each F_i into $U_k(I_i)$. We therefore get an ancestry mapping from all F_i's to $U_k(I)$, by Claim 3.2. It is now left to map the nodes in the spine S into $U_{k+1}(I)$, in a way that will respect the ancestry relation.

For every i, $1 \leq i \leq d'$, let $\widehat{h}_i = \sum_{j=1}^{i} \overline{h}_j$. We map the node v_i of the spine to the triplet $(k+1, h_1, \widehat{h}_i)$.

Let us now show that $(k+1, h_1, \widehat{h}_i)$ is in $U_{k+1}(I)$. First, the fact that $I_{k+1,h_1,\widehat{h}_i} \subseteq I$ follows from the fact that $I_{k+1,h_1,\widehat{h}_i} = \cup_{j=1}^{i} I_j$, and using $\bigcup_{j=1}^{d'} I_j \subseteq I$. It remains to show that $h_1 \in [0, H_{k+1})$ and that $\widehat{h}_i \in [0, J_{k+1})$. Note that,

$$a < 3n \leq \frac{3nd(k+1)^2}{2^k} \left\lceil \frac{2^k}{d(k+1)^2} \right\rceil = (H_{k+1} - 1)x_{k+1}.$$

Therefore, the smallest integer h_1 such that $a \leq h_1 x_{k+1}$ must satisfy $h_1 \in [0, H_{k+1})$. Recall now that for every i, $1 \leq i \leq d'$, $\bar{h}_i$ is the smallest integer such that $\lfloor c_k|F_i| \rfloor \leq \bar{h}_i x_{k+1}$. Thus

$$\bar{h}_i - 1 < \frac{\lfloor c_k|F_i| \rfloor}{x_{k+1}}.$$

It follows that,

$$\sum_{j=1}^{i} (\bar{h}_j - 1) < \sum_{j=1}^{i} \frac{\lfloor c_k|F_j| \rfloor}{x_{k+1}} \leq \frac{\lfloor c_k|F| \rfloor}{x_{k+1}} \leq \frac{c_k 2^{k+1}}{x_{k+1}} \leq 2c_k d(k+1)^2.$$

Thus

$$\sum_{j=1}^{i} \bar{h}_j < d + 2c_k d(k+1)^2 \leq 2c_{k+1} d(k+1)^2 = J_{k+1}.$$

Therefore $\widehat{h}_i \in [0, J_{k+1})$.

We now show that our mapping is indeed an ancestry mapping. Observe first that, for i and j such that $1 \leq i < j \leq d'$, we have

$$I_{k+1,h_1,\widehat{h}_i} \subset I_{k+1,h_1,\widehat{h}_j}.$$

Thus, the interval associated with v_j contains the one associated with v_i, as desired.

In addition, recall that, for every $i = 1, \ldots, d'$, F_i is mapped into $U_k(I_i)$. Therefore, if $L(v)$ is the triplet of some node $v \in F_i$, then the interval associated with it is contained in I_i. Since $I_i \subset I_{k+1,h_1,\widehat{h}_j}$ for every j such that $1 \leq i < j \leq d'$, we obtain that the interval associated with v is contained in the interval associated with v_j. This completes the proof of Claim 3.3.

From Claim 3.3, we get that there exists an ancestry mapping of any $F \in \mathcal{F}(n,d)$ into U. We use this ancestry mapping to label the nodes in F: an ancestry query between two labels can be answered using a simple interval containment test between the corresponding intervals. The stated label size follows, as each node can be encoded using $\log |U|$ bits, and $|U| = \Gamma_{\log n} = O(nd^2)$. This completes the proof of the theorem. □

4 A compact adjacency labeling scheme and a small universal graph for $\mathcal{F}(n,d)$

The ancestry labeling scheme described in the previous section can be advantageously transformed into an adjacency labeling scheme for trees of bounded depth. Recall that an *adjacency labeling scheme* for the family of graphs $\mathcal{G}$ is a pair $(\mathcal{M},\mathcal{D})$ of marker and decoder, satisfying that if $L(u)$ and $L(v)$ are the labels given by the marker $\mathcal{M}$ to two nodes u and v in some graph $G \in \mathcal{G}$, then

$$\mathcal{D}(L(u),L(v)) = 1 \iff u \text{ and } v \text{ are adjacent in } G.$$

Similarly to the ancestry case, we evaluate an adjacency labeling scheme $(\mathcal{M},\mathcal{D})$ by its *label size*, namely the maximum number of bits in a label assigned by the marker algorithm $\mathcal{M}$ to any node in any graph in $\mathcal{G}$.

For any two nodes u and v in a rooted forest F, u is a parent of v if and only if u is an ancestor of v and $depth(u) = depth(v) - 1$. Also, u is a neighbor of v if and only if either u is a parent of v or v is a parent of u. It therefore follows that one can easily transform any ancestry labeling scheme for $\mathcal{F}(n,d)$ to an adjacency labeling scheme for $\mathcal{F}(n,d)$ with an extra additive term of $\log d$ bits to the label size (these bits are simply used to encode the depth of a vertex). Using Theorem 3.1 we thus obtain the following.

Theorem 4.1 *There exists an adjacency labeling scheme for $\mathcal{F}(n,d)$ of size $\log n + 3\log d + O(1)$.*

Interestingly enough, this latter adjacency labeling scheme enables to give a short implicit representation (in the sense of [16]) of all forests with bounded depth. Recall that a graph G is an *induced subgraph* of a graph $\mathcal{U}$ if there exists a one-to-one (but not necessarily onto) mapping ϕ from $V(G)$ to $V(\mathcal{U})$ such that

$$\forall u,v \in V(G), \quad \{u,v\} \in E(G) \iff \{\phi(u),\phi(v)\} \in E(\mathcal{U}).$$

Given a graph family $\mathcal{G}$, a graph $\mathcal{U}$ is *universal* for $\mathcal{G}$ if every graph in $\mathcal{G}$ is an induced subgraph of $\mathcal{U}$. Note that a variant of this notion considers the graph $\mathcal{U}$ as universal for $\mathcal{G}$ whenever every graph in $\mathcal{G}$ is a partial subgraph of $\mathcal{U}$, i.e., the existence of an edge between $\phi(u)$ and $\phi(v)$ in $E(\mathcal{U})$ does not necessarily imply the existence of the edge $\{u,v\}$. This variant enables to analyze universal graphs for infinite graph classes [29]. The notion of universality considered in this paper is somewhat more restrictive, but it enables to relate the size of a universal graph for $\mathcal{G}$ with the size of the graphs in $\mathcal{G}$. Moreover, this notion of universality precisely captures the structure of the graphs in $\mathcal{G}$. In fact, there is a tight relation between this notion and adjacency labeling schemes:

Lemma 4.2 (S. Kannan, M. Naor, and S. Rudich [16])
A graph family $\mathcal{G}$ has an adjacency labeling scheme with label size k if and only if there exists a universal graph for $\mathcal{G}$, with 2^k nodes.

Combining the lemma above with Theorem 4.1, we get the corollary below.

Corollary 4.3 *Let d be a constant integer. There exists a universal graph for $\mathcal{F}(n,d)$, with $O(n)$ nodes.*

Proving or disproving the existence of a universal graph with a linear number of nodes for the class of n-node trees is a central open problem in the design of informative labeling schemes.

References

[1] S. Abiteboul, S. Alstrup, H. Kaplan, T. Milo and T. Rauhe. Compact labeling schemes for ancestor queries. *SIAM Journal on Computing* **35**, (2006), 1295–1309.

[2] S. Abiteboul, P. Buneman and D. Suciu. Data on the Web: From Relations to Semistructured Data and XML. *Morgan Kaufmann*, (1999).

[3] Abiteboul, S., Kaplan, H., and Milo, T.: Compact labeling schemes for ancestor queries. In: Proc. 12th ACM-SIAM Symp. on Discrete Algorithms, (2001).

[4] S. Alstrup, P. Bille and T. Rauhe. Labeling Schemes for Small Distances in Trees. *SIAM J. Discrete Math* **19(2)**, (2005), 448–462.

[5] S. Alstrup, C. Gavoille, H. Kaplan and T. Rauhe. Nearest Common Ancestors: A Survey and a new Distributed Algorithm. *Theory of Computing Systems* **37**, (2004), 441–456.

[6] S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In *Proc. 43'rd annual IEEE Symp. on Foundations of Computer Science*, Nov. 2002.

[7] Cohen, E., Kaplan, H., and Milo, T.: Labeling dynamic XML trees. In *Proc. 21st ACM Symp. on Principles of Database Systems*, (2002).

[8] A. Deutsch, M. Fernndez, D. Florescu, A. Levy and D. Suciu. A Query Language for XML. *Computer Networks* **31**, (1999), 1155-1169.

[9] P. Fraigniaud and C. Gavoille. Routing in trees. In *Proc. 28th Int. Colloq. on Automata, Languages & Prog.*, LNCS 2076, pages 757–772, July 2001.

[10] C. Gavoille and C. Paul. Split decomposition and distance labelling: an optimal scheme for distance hereditary graphs. In *Proc. European Conf. on Combinatorics, Graph Theory and Applications*, Sept. 2001.

[11] C. Gavoille and D. Peleg. Compact and Localized Distributed Data Structures. *J. of Distributed Computing* **16**, (2003), 111–120.

[12] C. Gavoille, D. Peleg, S. Pérennes and R. Raz. Distance labeling in graphs. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, pages 210–219, Jan. 2001.

[13] C. Gavoille, M. Katz, N.A. Katz, C. Paul and D. Peleg. Approximate Distance Labeling Schemes. In *9th European Symp. on Algorithms*, Aug. 2001, Aarhus, Denmark, SV-LNCS 2161, 476–488.

[14] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *Workshop on Algorithms and Data Structures*, Aug. 2001.

[15] H. Kaplan, T. Milo and R. Shabo. A Comparison of Labeling Schemes for Ancestor Queries. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002.

[16] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In *SIAM J. on Descrete Math* **5**, (1992), 596–603.

[17] M. Katz, N.A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. *SIAM Journal on Computing* **34** (2004),23–40.

[18] A. Korman. General Compact Labeling Schemes for Dynamic Trees. J. Distributed Computing 20(3): 179-193 (2007).

[19] A. Korman. Labeling Schemes for Vertex Connectivity. ACM Transactions on Algorithms, to appear.

[20] A. Korman. Improved Compact Routing Schemes for Dynamic Trees In *Proc. 27th Ann. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing (PODC)*, 2008.

[21] A. Korman and S. Kutten. Distributed Verification of Minimum Spanning Trees. J. Distributed Computing 20(4): 253-266 (2007).

[22] A. Korman and D. Peleg. Labeling Schemes for Weighted Dynamic Trees. J. Information and Computation 205(12): 1721-1740 (2007).

[23] A. Korman, D. Peleg, and Y. Rodeh. Labeling schemes for dynamic tree networks. *Theory of Computing Systems* **37** (2004), pp. 49-75.

[24] A. Korman, D. Peleg, and Y. Rodeh. Constructing Labeling Schemes Through Universal Matrices. *Algorithmica*, to appear.

[25] L. Mignet, D. Barbosa and P. Veltri. Studying the XML Web: Gathering Statistics from an XML Sample. *World Wide Web* **8(4)**, (2005), 413–438.

[26] I. Mlynkova, K. Toman and J. Pokorny. Statistical Analysis of Real XML Data Collections. In *Proc. 13th Int. Conf. on Management of Data*, (2006), 20 – 31.

[27] D. Peleg. Proximity-preserving labeling schemes and their applications. In *Proc. 25th Int. Workshop on Graph-Theoretic Concepts in Computer Science*, pages 30–41, June 1999.

[28] D. Peleg. Informative labeling schemes for graphs. In *Proc. 25th Symp. on Mathematical Foundations of Computer Science*, volume LNCS-1893, pages 579–588. Springer-Verlag, Aug. 2000.

[29] R. Rado. Universal graphs and universal functions. Acta Arithmetica **9**:331-340, 1964.

[30] N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The Computer Journal* **28**, (1985), 5–8.

[31] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. of the ACM* **51**, (2004), 993–1024.

[32] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. on Parallel Algorithms and Architecture (SPAA)*, pages 1–10, Hersonissos, Crete, Greece, July 2001.

[33] W3C.Extensive markup language (XML) 1.0. http://www.w3.org/TR/REC-xml.

[34] W3C. Exensive stylesheet language (xsl) 1.0. http://www.w3.org/Style/XSL/.

[35] W3C. Xsl transformations (xslt) specification. http://www.w3.org/TR/WD-xslt